C# 프로그래밍

- 2 주차 (1강)

SEJONG UNIVERSITY

데이터 타입

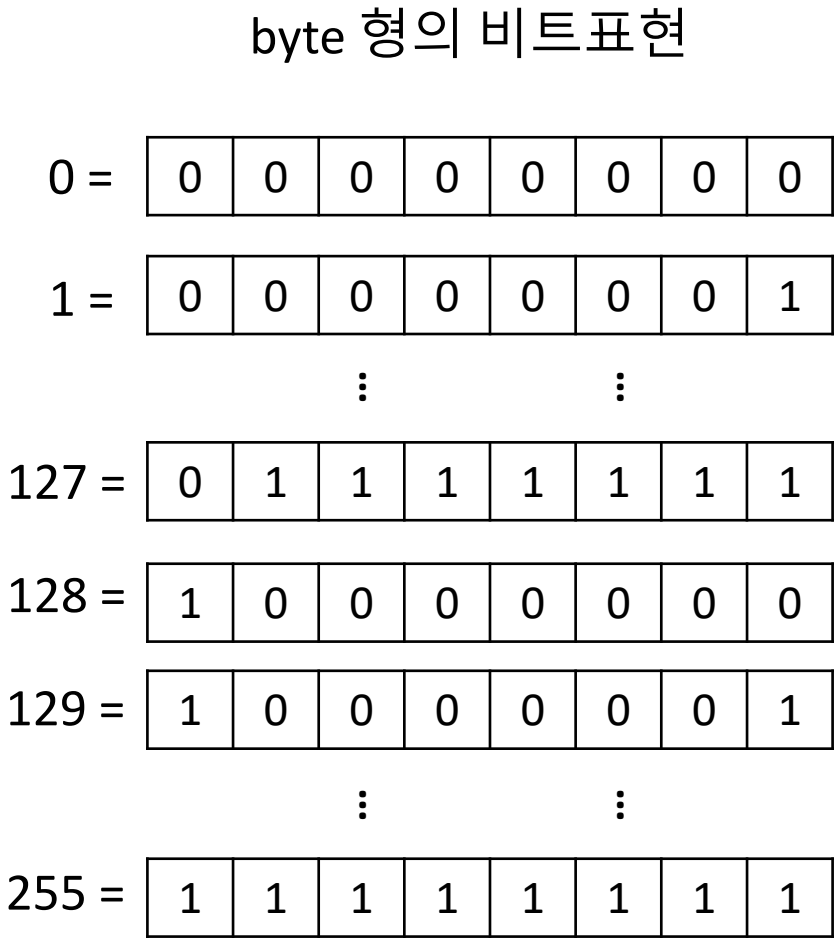
- 기본 데이터 형식
- 데이터형 바꾸기
- 상수와 열거 형식
- Nullable 형식과 var 키워드
- CTS 표준 데이터 형

기본 데이터 형식

- 부호 있는, 부호 없는 정수 형식
- 부동 소수점 형식
- 문자 형식과 논리 형식
- object 형식

부호 있는 정수와 부호 없는 정수

- 부호 있는 정수형
  - ✓ 음수(-), 0, 양수(+)
  - ✓ Sbyte, short, int, long
- 부호 없는 정수형
  - ✓ 0, 양수(+)
  - ✓ byte, ushort, uint, ulong
- byte와 sbyte 비교
  - ✓ 두 형식 모두 1 바이트, 즉 8 비트로 구성
  - ✓ byte는 1~255 표현, sbyte는 -128~127 표현

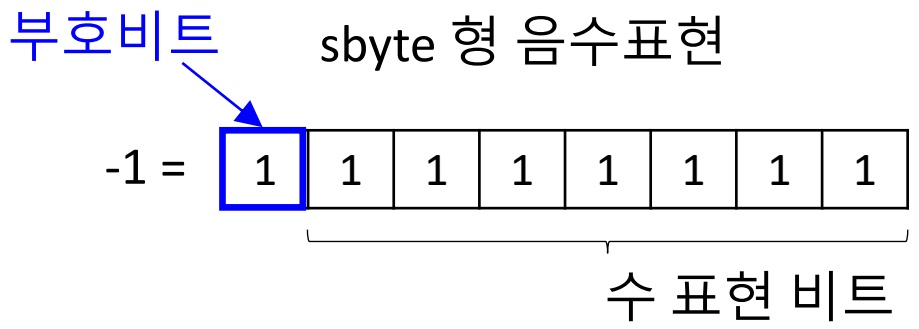


sbyte 형의 음수 표현 : 2의 보수

- sbyte의 비트표현
  - ✓ 0과 양수의 비트표현은 byte와 동일
  - ✓ 음수 비트표현은 2의 보수법 이용

sbyte 형 0과 양수표현

0 =	0	0	0	0	0	0	0	0
1 =	0	0	0	0	0	0	0	1
	⋮				⋮			
127 =	0	1	1	1	1	1	1	1



2의 보수법 (예: -1 표현)

(1) 1을 수 표현 비트에 입력

0	0	0	0	0	0	0	1

(2) 1은 0으로 0은 1로 반전

1	1	1	1	1	1	1	0

(3) 반전된 비트에 1을 더함

1	1	1	1	1	1	1	1

예제 코드

```
static void Main(string[] args)
{
    byte a = 255;         // 비트표현: 1111 1111
    sbyte b = (sbyte)a;

    byte c = 128;         // 비트표현: 1000 0000
    sbyte d = (sbyte)c; // sbyte의 범위 -128 ~ 127

    Console.WriteLine(a);
    Console.WriteLine(b);
    Console.WriteLine(c);
    Console.WriteLine(d);
}
```

출력 결과:

```
255
-1
128
-128
```

오버플로 (Overflow) 예제 코드

```
static void Main(string[] args)
{
    uint a = uint.MaxValue; // uint형 최대값 4294967295
    Console.WriteLine(a);
    a = a + 1;              // 0000 0000 … … 0000
    Console.WriteLine(a);

    int b = int.MaxValue;   // int형 최대값 2147483647
    Console.WriteLine(b);
    b = b + 1;              // 1000 0000 … … 0000
    Console.WriteLine(b);
}
```

출력 결과:
4294967295
0
2147483647
-2147483648

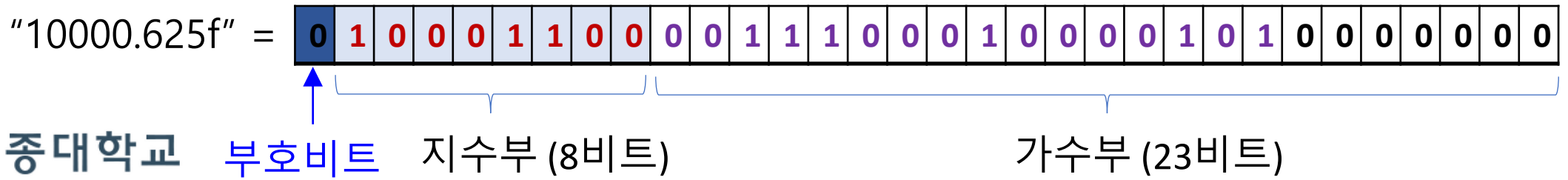
부동 소수점 형식

- 소수점이 고정되어 있지 않고 움직이면서 수를 표현
- 정수와 유리수 포함
  - ✓ 정수 데이터형과 지원하는 값의 범위가 다름
  - ✓ 산술 연산과정이 복잡
- 정밀도
  - ✓ float < double < decimal

데이터형	유효숫자	크기 (비트)	범위
float	7개	32	-3.402823e38 ~ 3.402823e38
double	15~16개	64	-1.79769313486232e308 ~ 1.79769313486232e308
decimal	29개	128	$\pm 1.0 \times 10e-28 \sim \pm 7.9 \times 10e28$

float 형 비트 표현 : IEEE754

- float과 double 형은 IEEE754 알고리즘 기반
- 예: float 형 데이터 "10000.625f"의 비트표현
  - ✓ 2진수로 변환 -> "10 0111 0001 0000.101"
  - ✓ 부동 소수점 형식으로 표현 -> "1.0011100010000101 x 2^13"
  - ✓ 지수부 8비트는 13+127=140 (즉, 2진수 "1000 1100")
  - ✓ 나머지 가수부는 소수점 이하자리 "00111000100001010"



부동 소수점 형식 예제 코드

```
static void Main(string[] args)
{
    // 숫자 뒤에 f를 붙이면 float 형식으로 간주
    float a = 3.1415_9265_3589_7932_3846_2643_3832f;
    // 숫자 뒤에 아무 것도 없으면 double 형
    double b = 3.1415_9265_3589_7932_3846_2643_3832;
    // 숫자 뒤에 m을 붙이면 decimal 형식으로 간주
    decimal c = 3.1415_9265_3589_7932_3846_2643_3832m;

    Console.WriteLine(a);
    Console.WriteLine(b);
    Console.WriteLine(c);
}
```

출력 결과:

```
3.1415927
3.141592653589793
3.14159265358979323846264338320
```

문자형식과 논리 형식

- char 형
  - ✓ 정수 계열의 형식이지만, '가' '나' 'a' 'b' 와 같은 문자 데이터를 다룸
  - ✓ ushort와 비트 상으로 16비트로 동일
  - ✓ 사칙연산 수행 시 컴파일 에러

```
char ch = 'A';
Console.WriteLine(ch);  // A 문자 출력
```

- 논리 형식
  - ✓ 참 또는 거짓 데이터를 다룸

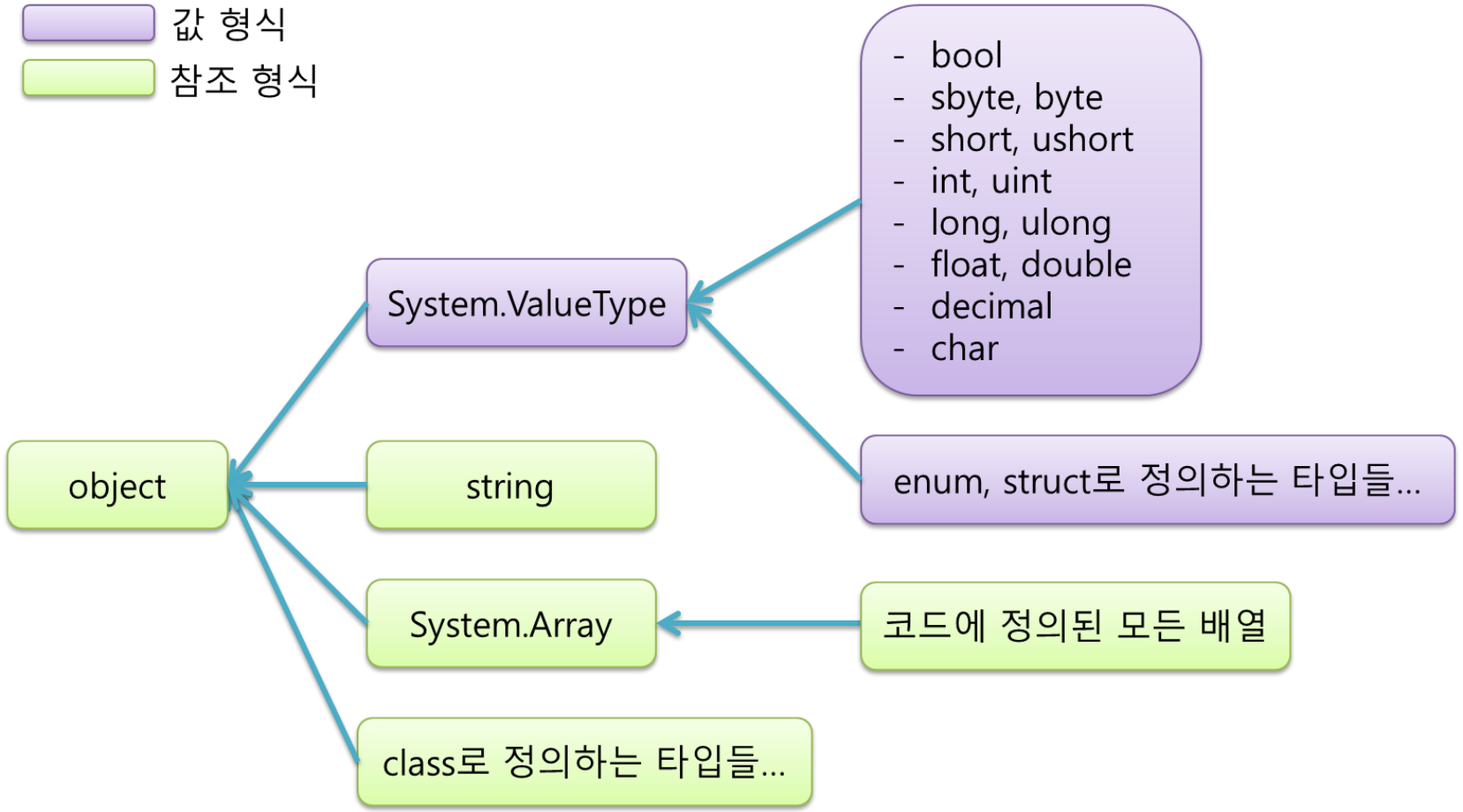
데이터형	설명	크기 (비트)	범위
bool	논리형식	8	true, false

```
bool a = true;
bool b = false;

Console.WriteLine(a);   // True 출력
Console.WriteLine(b);   // False 출력
```

object 형식

- 어떤 데이터이든지 다룰 수 있는 형식
  - ✓ 모든 데이터 형식이 object 형식으로부터 상속 받음



object 형식 예제 코드

- 어떤 데이터이든지 다룰 수 있는 형식
  - ✓ int, long, char, bool 등 모든 형식이 object 형식으로부터 상속 받음

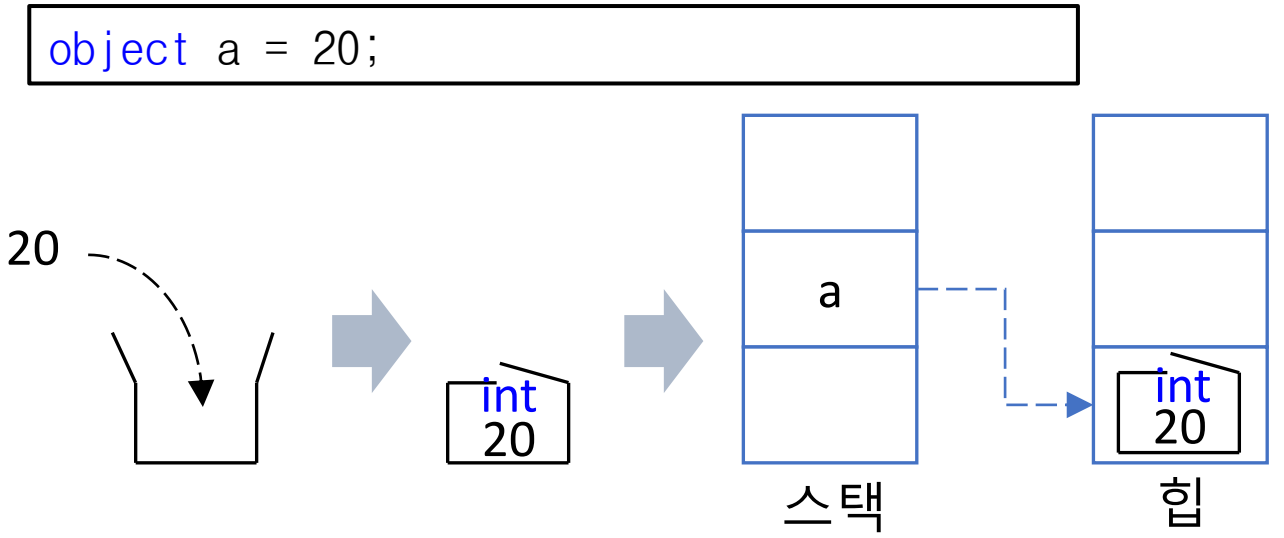
```
static void Main(string[] args)
{
    object a = 123;
    object b = 3.1415_9265_3589_7932_3846_2643_3832m;
    object c = true;
    object d = "안녕하세요.";

    Console.WriteLine(a);
    Console.WriteLine(b);
    Console.WriteLine(c);
    Console.WriteLine(d);
}
```

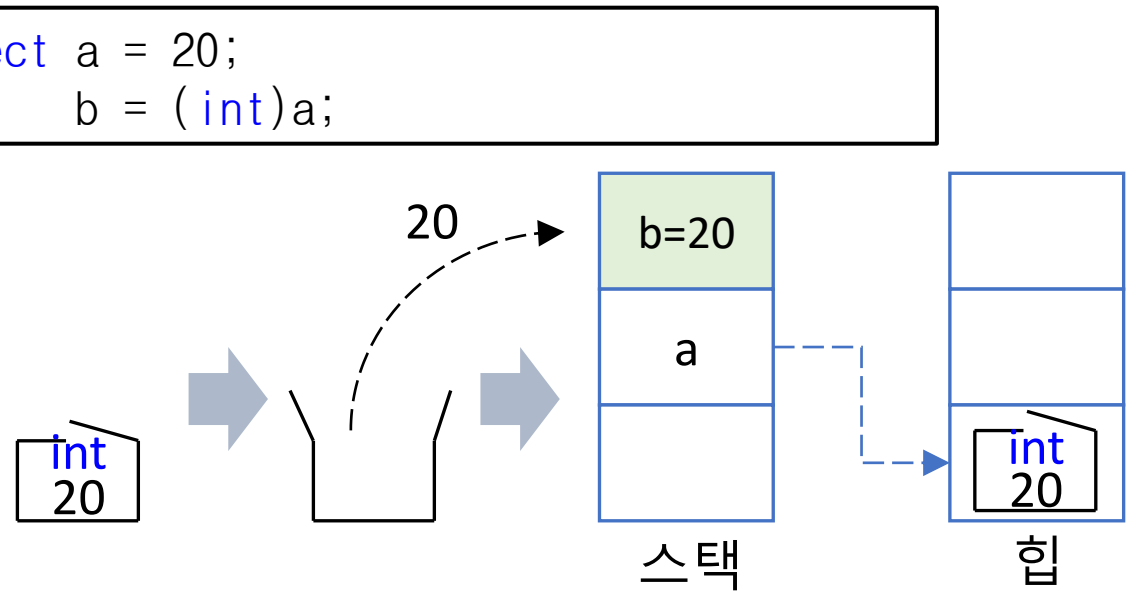
출력 결과:
123
3.14159265358979323846264338 32
True
안녕하세요

박싱과 언박싱

- 박싱 (Boxing)
  - ✓ 값 형식의 데이터를 힙에 할당 방법



- 언박싱 (Unboxing)
  - ✓ 박싱된 값을 꺼내 값 형식 변수에 저장하는 과정



데이터 형 바꾸기

SEJONG UNIVERSITY

데이터형 변환

- 암시적 변환
  - ✓ 범위가 작은 데이터 타입에서 범위가 큰 타입으로 변환
  - ✓ 부가적인 코드를 지정하지 않고 형변환

```
byte b = 250; // byte 범위 0~255
short s = b;  // short 범위 -32,768 ~ 32,767
```

- 명시적 변환
  - ✓ 개발자가 의도한 형변환임을 컴파일러에 알리는 변환
  - ✓ 예: 개발자가 의도하여 char 형 변수에 'A'의 숫자 값 65 입력

```
ushort  u = 65;          //  'A' 의 숫자 값은 65
char    c = (char)u;     // char 형에 숫자를 그대로 입력하면 에러 발생
Console.WriteLine(c);    // 출력 결과: A
```

크기가 서로 다른 정수형 사이의 변환

```
static void Main(string[] args)
{
    sbyte a = 127;
    Console.WriteLine(a);

    int b = a;      // 암시적 형변환
    Console.WriteLine(b);

    int x = 128;    // sbyte의 최대값 127보다 1 큰 수
    Console.WriteLine(x);

    sbyte y = (sbyte)x; // 명시적 형변환 후 오버플로 발생
    Console.WriteLine(y);
}
```

출력 결과:
127
127
128
-128

크기가 다른 부동 소수점 형식 사이 변환

- float 형과 double 형 모두 소수를 2진수로 메모리에 저장
- float 형과 double 형 사이의 변환
  - ✓ 메모리에 저장된 2진수를 10진수로 복원 후, 다시 2진수로 변환
  - ✓ 2진수로 모든 소수를 완벽 표현 불가능, 두 데이터형 사이 변환 시 정밀도에 손상 발생 가능

```
float c = 0.1f;
Console.WriteLine(c);

double d = (double)c;
Console.WriteLine(d);
```

출력 결과:
0.1
0.10000000149011612

부호 있는, 부호 없는 정수형 사이 변환

- 변환에 참여하는 두 정수형의 값 범위를 유의
  - ✓ 변환 전, 오버플로 또는 언더플로 발생하는지 확인 필요

```
static void Main(string[] args)
{
    int a = 500;
    Console.WriteLine(a);

    uint b = (uint)a;
    Console.WriteLine(b);

    int x = -30;
    Console.WriteLine(x);

    uint y = (uint)x;   // 언더플로 발생
    Console.WriteLine(y);
}
```

출력 결과:
500
500
-30
4294967266

부동 소수점형에서 정수형으로 변환

- 부동 소수점형 변수는 정수형 범위 내의 값을 사용
- 정수형으로 변환 시 소수점 이하 버림

```
static void Main(string[] args)
{
    float a = 0.9f;
    int b = (int)a;
    Console.WriteLine(b);

    float c = 1.1f;
    int d = (int)c;
    Console.WriteLine(d);
}
```

```
출력 결과:
0
1
```

문자열과 숫자 사이의 변환

- 명시적 변환 시 컴파일 에러 발생

```
string a = "12345";
int b = (int)a; // 컴파일 에러
```

```
int c = 12345;
string d = (string)c; // 컴파일 에러
```

- "Parse (·)" 메소드
  - ✓ 숫자로 반환할 문자열을 숫자로 변환
  - ✓ 모든 정수, 부동 소수점 형식에서 지원되는 메소드
- "ToString ()" 메소드
  - ✓ 숫자로부터 문자열 출력
  - ✓ object로부터 상속 받은 ToString() 메소드를 재정의

문자열과 숫자 사이의 변환: 예제 코드

```
static void Main(string[] args)
{
    int a = 123;
    string b = a.ToString();
    Console.WriteLine(b);

    float c = 3.14f;
    string d = c.ToString();
    Console.WriteLine(d);

    string e = "123456";
    int f = int.Parse(e);
    Console.WriteLine(f);

    string g = "1.2345";
    float h = float.Parse(g);
    Console.WriteLine(h);
}
```

출력 결과:

```
123
3.14
123456
1.2345
```

상수와 열거 형식

SEJONG UNIVERSITY

상수와 열거 형식

- 상수와 열거 형식은 안에 담긴 데이터를 변경 불가
- 상수의 선언은 변수의 선언과 유사

선언 형식

```
const 자료형 상수명 = 값;
```

선언 예:

```
const int a = 3;
const double b = 3.14;
const string c = "abcdef";

a = 4; // 상수 값 변경 시 컴파일 에러 발생
```

- 열거 형식은 여러 개의 상수를 선언

선언 형식

```
enum 열거형식명 : 기반자료형 {상수1, 상수2, …}
```

선언 예:

```
enum DialogResult {YES, NO, CANCEL, CONFIRM, OK}
```

열거 형식

선언 형식

```
enum 열거형식명 : 기반자료형 {상수1, 상수2, …}
```

선언 예:

```
enum DialogResult {YES, NO, CANCEL, CONFIRM, OK}
```

- 기반 자료형은 정수 계열
  - ✓ byte, sbyte, short, ushort, int, uint, long, ulong, char
  - ✓ 생략할 경우 컴파일러는 int 자료형 사용
- 열거 형식 안에 선언된 상수에 값 할당이 없는 경우
  - ✓ 각 상수에 0부터 자동으로 1씩 증가하여 차례로 할당

```
static void Main(string[] args)
{
    Console.WriteLine((int)DialogResult.YES);
    Console.WriteLine((int)DialogResult.NO);
    Console.WriteLine((int)DialogResult.CANCEL);
    Console.WriteLine((int)DialogResult.CONFIRM);
    Console.WriteLine((int)DialogResult.OK);
}
```

출력 결과:

```
0
1
2
3
4
```

열거 형식 상수 값 할당

- 열거형식 선언 시 개발자가 상수 값 할당 가능
  - ✓ 값이 할당 되지 않은 상수는 앞 상수 값에 1일 더한 값
  - ✓ 맨 앞 상수가 할당되지 않은 경우 0으로 자동 할당

```
enum DialogResult {YES, NO, CANCEL = 50, CONFIRM, OK}
```

```
static void Main(string[] args)
{
    Console.WriteLine((int)DialogResult.YES);
    Console.WriteLine((int)DialogResult.NO);
    Console.WriteLine((int)DialogResult.CANCEL);
    Console.WriteLine((int)DialogResult.CONFIRM);
    Console.WriteLine((int)DialogResult.OK);
}
```

출력 결과:
0
1
50
51
52

열거 형식의 변수 생성

- 열거 형식의 선언은 새로운 데이터형식의 생성 의미
  - ✓ 선언된 열거형식을 통해 변수 생성 가능
  - ✓ 변수에는 열거형식에서 정의 된 상수만 담을 수 있음

```
enum DialogResult { YES, NO, CANCEL, CONFIRM, OK}
static void Main(string[] args)
{
    DialogResult result = DialogResult.YES;

    Console.WriteLine(result == DialogResult.YES);
    Console.WriteLine(result == DialogResult.NO);
    Console.WriteLine(result == DialogResult.CANCEL);
    Console.WriteLine(result == DialogResult.CONFIRM);
    Console.WriteLine(result == DialogResult.OK);
}
```

출력 결과:

```
True
False
False
False
False
```

Nullable 형식과 var 키워드

Nullable 형식

- 값 형식의 변수가 "비어 상태가" 될 수 있는 형식
- 원래 데이터 형에 '?'를 붙여서 선언

선언 형식

```
데이터형식? 변수이름;
```

선언 예:

```
int?    a = null;
float?  b = null;
double? c = null;
```

- Nullable 형식 변수에 값 할당 없이 사용하면 컴파일 에러 발생
  - ✓ null 또는 해당 데이터 형 범위 내의 값 할당 가능
- HasValue와 Value 속성을 가짐
  - ✓ HasValue: 해당 변수가 값을 가지고 있는지 확인
  - ✓ Value: 변수에 담겨 있는 값

HasValue와 Value 예제 코드

```
static void Main(string[] args)
{
    int? a = null;
    Console.WriteLine(a.HasValue);
    Console.WriteLine(a != null);
    //Console.WriteLine(a.Value); 런타임 에러 발생 (null 할당되어 있음)

    a = 3;
    Console.WriteLine(a.HasValue);
    Console.WriteLine(a != null);
    Console.WriteLine(a.Value);

    //int? b;
    //Console.WriteLine(b.HasValue); 컴파일 에러 발생 (초기화 필요)
}
```

출력 결과:

```
False
False
True
True
3
```

var 키워드

- 변수를 선언하면 컴파일러가 자동으로 해당 변수의 형식 지정
  - ✓ 변수 선언과 동시에 초기화 필요
  - ✓ 초기화 데이터에 따라 해당 변수의 형식 지정
  - ✓ 지역 변수로만 사용할 수 있음

```
var a = 3;          // a는 int 형식
var b = "Hello"; // b는 string 형식
```

```
object a = 3;
object b = "Hello";
```

- var 키워드와 object 형식과의 차이점
  - ✓ 예를 들어, "var a = 3;"과 "object a = 3;"를 비교하면,
  - ✓ object 형식의 경우 CLR이 3을 박싱해서 힙에 넣고 a 가 힙을 가리키도록 설정
  - ✓ var 키워드의 경우 컴파일 시점에서 a의 데이터형을 int로 바꿔 컴파일

CTS 표준 데이터형

SEJONG UNIVERSITY

CTS의 데이터형식

- 공용형식 시스템 (CTS)
  - ✓ 모든 .NET 호환 언어들이 따르는 데이터 형식 표준
  - ✓ CTS에서 정의하는 데이터형식을 모두 구현할 필요는 없음
  - ✓ C#의 데이터 형식 체계가 CTS 표준을 따름
  - ✓ CTS의 데이터형식은 코드에서 그대로 사용 가능

클래스 이름	C# 형식	C++ 형식	VB 형식
System.Byte	byte	unsigned char	Byte
System.SByte	sbyte	char	SByte
System.Int16	short	short	Short
System.Int32	int	int	Integer
System.Int64	long	__int64	Long
System.UInt16	ushort	unsigned short	UShort
System.UInt32	uint	unsigned int	UInteger
System.UInt64	ulong	unsigned __int64	ULong
System.Single	float	float	Single
System.Double	double	double	Double
System.Boolean	bool	bool	Boolean
System.Char	char	wchar_t	Char
System.Decimal	decimal	Decimal	Decimal
System.IntPtr	없음	없음	없음
System.UIntPtr	없음	없음	없음
System.Object	object	Object*	Object
System.String	string	String*	String

CTS 데이터형식 예제 코드

```
static void Main(string[] args)
{
    System.Int32 a = 123;
    int b = 456;
    Console.WriteLine("a type:{0}, value:{1}", a.GetType(), a);
    Console.WriteLine("b type:{0}, value:{1}", b.GetType(), b);

    System.String c = "abc";
    string d = "def";
    Console.WriteLine("c type:{0}, value:{1}", c.GetType(), c);
    Console.WriteLine("d type:{0}, value:{1}", d.GetType(), d);
}
```

출력 결과:

```
a type: System.Int32, value: 123
b type: System.Int32, value: 456
c type: System.String, value: abc
d type: System.String, value: def
```

Thank you!